



26K1
6-6-94

37⁰⁰ 203 CP
#19/Supplemental 268
Arnold D. Appendix
R. Morgan
6/10/94
780.29643X00

THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicants: Thomas J. CAMPANA, JR. et al
Serial No.: 07/702,939
Filed: May 20, 1991
For: ELECTRONIC MAIL SYSTEM WITH RF
COMMUNICATIONS TO MOBILE PROCESSORS
Group: 2608
Examiner: G. Oehling

RECEIVED
91 JUN -7 PM 3:56
GROUP 260

THIRD SUPPLEMENTAL AMENDMENT

Honorable Commissioner of
Patents and Trademarks
Washington, D. C. 20231

May 23, 1994

Sir:

This is supplemental to the Second Supplemental Amendment
filed May 13, 1994.

IN THE SPECIFICATION:

Please amend the specification as follows:

Page i (before Page 1), line 12, change "10-14" to
--10-12--.

In the Attached Appendix: Delete the original 15 pages
of the Appendix as filed on May 20, 1991 and insert a
substitute Appendix which is attached to this Third
Supplemental Amendment consisting of a cover page and numbered
pages 1-12.

120 WP 06/06/94 07702939

1 203

37.00 CK

IN THE CLAIMS:

Please add new claim 176 as follows:

--176. A method of transferring information from a RF receiver to a processor under control of a program stored by the processor comprising:

transmitting the information with a RF transmitter to the RF receiver;

the RF receiver signalling the processor on a transmission medium of the processor used for transmission of information by the processor that the received information is stored within memory of the receiver;

controlling the transfer of the stored information from the memory of the receiver to a memory of the processor on the transmission medium with the program; and

processing the information in the memory of the processor with an application program stored in the memory of the processor.--

REMARKS

The Applicants are presenting claim 176 herein which is identical to claim 81 of Serial No. 07/702,319 and claim 86 of Serial No. 07/702,938 for the purpose of having the Examiner consider the possibility of double patenting between claims 78, 79 and 81-89 in Serial No. 07/702,319 and claims 68, 69 and 86 in Serial No. 07/702,938.

If the Examiner finds the subject matter of at least claim 176 to be present a situation requiring restriction, it is requested that the Examiner indicate in this application on the record to that effect and in Serial Nos. 07/702,319 and 07/702,938, respectively, with respect to claims 78, 79 and 81-89 and 68, 69 and 86. If restriction is required in this application and in Serial Nos. 07/702,319 and 07/702,938 regarding at least the subject matter of claim 176, Applicants will cancel the subject matter so restricted from all applications now pending and file a divisional application.

Newly submitted claim 176 covers the operation of the receiver 119 transferring originated information to the destination processors, as illustrated in Fig. 10, as described on pages 46 and 47 of the specification and pages 1-9 of the Appendix. Claims 160 and 161, which are respectively dependent on claims 137 and 157, also recite aspects of the transfer of the other originated information for use with application programs.

The Examiner, for purposes of determining the patentability of claim 176, is referred to the description of the prior art including the paging receiver(s) 119 and peripheral device 119' in Figs. 2 and 7 as described in the specification. Further, a description of a connection of a paging receiver to a peripheral device as prior art is found in the United States Patents and pending applications identified on page 9 of the specification.

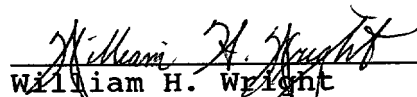
The substitute Appendix contains numbered pages which are consistent with the description of the page numbers of the Appendix on page i of the specification as amended. The copyright notices on pages 4 and 10 of the original Appendix have been deleted from pages 4 and 10 of the substitute Appendix to be consistent with the copyright notice which precedes page 1 of the original and substitute Appendix. The description of the Appendix on page i of the specification has been amended to refer to pages 10-12 as being the program for controlling the operation of the interface switch. Deleted pages 13-14 were not used as the code for controlling the interface switch.

The inventors will file a Supplemental Declaration affirming subject matter of this Amendment as being part of their invention as filed.

Please charge any shortage in the fees due in connection with the filing of this paper, including extension of time fees, to the deposit account of Antonelli, Terry, Stout & Kraus, Deposit Account No. 01-2135 (780.29767X00), and please credit any excess fees to such deposit account.

Respectfully submitted,

HENDERSON & STURM



William H. Wright
Registration No. 26,424

WHW:dlh


```

#include <string.h>
#include <time.h>
#include <stdio.h>
#include <dos.h>
#include "safer1.h"

void main(void)
{
    FILE *infile,*outfile;
    char buffer[81],chr,timestr[6],datestr[9];
    char msg_num[4];
    int msg_num_opt = 0;
    char *ptr;
    int x,day,month,line=1,attmail=0;
    time_t t;

    if ((infile = fopen(ATT_EMAIL_FILE,"rt")) == NULL)
    {
        printf("%s does not exist\n",ATT_EMAIL_FILE);
        exit(0);
    }
    if ((outfile = fopen("tfmbox.888","wt")) == NULL)
    {
        printf("Can't open TFMBOX.888\n");
        exit(0);
    }

    for(;;)
    {
        /* get characters from .tmp file */
        x = 0;
        do
        {
            chr = fgetc(infile);
            if (feof(infile))
            {
                fclose(infile);
                fclose(outfile);
                exit(0);
            }
            buffer[x++] = chr;
        }
        /* until end of line */
        while (chr != '\n' && x != 80);

        buffer[x] = '\0'; /* terminate it */

        if (line == 1)
        {
            ptr = strchr(buffer,');');
            if (ptr-buffer == 2) /* use 3rd character */
            {
                ascanf(buffer,"%i")&msg_num;
                msg_num_opt = 1;
                ptr++;
            }
            else
                ptr = buffer;

            if (*ptr == ':' && *(ptr+1) == 'D')
                attmail = 1;
        }

        if (attmail)
        {
            switch(line)

```


TELEFIND CORP.

```

/*
Copyright:      1990 TELEFIND CORP.
Author:         MICHAEL P. PONSCHKE, BR.
                03/13/91

Program:        SAFARI3.C
Purpose:        TO EXTRACT MESSAGES FROM A TELEFIND PAGER
                VIA IN RS-232 PORT ON A PC

Compiler:       TURBO C++ 1.0
Memory Model:   SMALL
*/

```

```

#include <dos.h>
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <stdlib.h>
#include "safari.h"

```

```

/*          CONSTANTS          */

```

```

#define DTR_HI      0x01
#define DTR_LO      0xfe
#define RTS_HI      0x02
#define RTS_LO      0xfd
#define DSR_HI      0x20
#define RING_IN     0x40
#define CD_HI       0x80
#define FIVE_TICK   5
#define FIVE_SEC    96
#define TWELVE_SEC  220
#define LOG_FILE    "LOG"
#define INTRO_STRING "Please standby, retrieving messages ..."

```

```

/*          FUNCTION PROTOTYPES          */

```

```

int beep(void);
void busyoff(void);
void busyon(void);
void dleoff(void);
void dleon(void);
int link(void);
void print_message(void);
int rxdata(void);
int strobe(void);
int strobe_data(void);
unsigned ticks(void);
int timeout(unsigned start, int delay);

```

```

/*          VARIABLE DECLARATIONS          */

```

```

char pager_buffer[511];
int com_base, control_reg, status_reg, log_flag;
FILE *log_file;

```

```

void main(int num_arg, char **args)
{

```

```

    unsigned start;
    int restart, x;

```

```

    com_base = 0x3f8; /* use com 1 unless command line denotes otherwise */

```

```

    /* get command line arguments */

```


TE0220 065704A

```

        fclose(log_file);
    }

    /*      pager beep      */
    int beep(void)
    {
        /*      accesses the RI line via the Status Register
            which is activated when the pager beeps      */
        unsigned start;

        start = ticks();
        while ( ! timeout(start,FIVE_TICK))
        {
            if ((inportb(status_reg) & RING_IN) == 0 )
                return(1);
        }
        return(0);
    }

    /*      busyon & busyoff toggle the DTR line via the
        Control Register to strobe in data from the pager      */
    void busyoff(void)
    {
        outportb(control_reg,inportb(control_reg) | DTR_HI);
    }

    void busyon(void)
    {
        outportb(control_reg,inportb(control_reg) & DTR_LO);
    }

    /*      dison & disoff toggle the RTS line via the Control Register
        to simulate the pressing of the display button on the pager      */
    void dison(void)
    {
        outportb(control_reg,inportb(control_reg) | RTS_HI);
    }

    void disoff(void)
    {
        outportb(control_reg,inportb(control_reg) & RTS_LO);
    }

    int link(void)
    {
        /*      accesses the CD line via the Status Register
            which is logic high when pager is connected      */
        if ((inportb(status_reg) & CD_HI) == 0)
            return(0);
        return(1);
    }

    void print_message(void)
    {
        FILE *file;
        unsigned start;
        int x,y=0,z=0,chr,bit;
    }

```

76240 6660 44

```
busyoff(); /* ready to accept pager data */
/* read until end code received */
while (chr != 3)
{
    chr = 0;
    start = ticks();

    /* wait for start bit */
    do
    {
        bit = strobe();
        if (bit == 0)
            break;
    }
    while (!timeout(start, FIVE_SEC));

    if (bit)
    {
        if (log_flag)
            fprintf(log_file, "Transmission Error, recheck connection\n");
        disoff();
        exit(0);
    }

    /* strobe out 8 bit data */
    for (x=1; x<9; x++)
    {
        chr <<= 1;
        chr |= bit = strobe_data();
    }

    /* clear out stop bits */
    for (x=1; x<3; x++)
    {
        strobe_data();
    }

    /* extract start and end codes from message
    pager signon      02, 18, 00, 33
    pager signoff     03 */

    if ((y > 3) && (chr != 3))
    {
        /* pager characters 96 and 97 are converted to
        0xFA and 0xFB to display on pager */
        if (chr == 0xFA) /* convert to CR */
            chr = '\n';
        if (chr == 0xFB) /* convert to TAB */
            chr = 0x09;

        pager_buffer[z] = chr;
        z++;
    }
    y++;
}

pager_buffer[z] = '\0'; /* null terminate */
busyon(); /* finished receiving data */
```

03702330 00001

```

if (log_file)
    fprintf(log_file, "%s\n", pager_buffer);

if ((file = fopen(ATT_EMAIL_FILE, "at")) == NULL)
    fprintf(log_file, "Unable to open TFMBOX.TMP\n");
else
{
    fprintf(file, "%s\n", pager_buffer);
    fprintf(file, "%s", DELIMITER);
    fclose(file);
}

start = ticks();
while(!timeout(start, FIVE_SEC))
{
    /* wait for erase beep */
    if (beep()) break;
}
sleep(1); /* wait one more second */
}

int rxdata(void)
{
    /* accesses the DSR line via the Status Register
    which returns the bits value */

    if (!inportb(status_reg) & DSR_M1)
        return(0);
    return(1);
}

int strobe(void)
{
    int bit;

    busyon();
    delay(1);
    busyoff();
    delay(4);
    bit = rxdata();
    return(bit);
}

int strobe_data(void)
{
    int bit;

    busyon();
    delay(2);
    bit = rxdata();
    busyoff();
    delay(1);
    return(bit);
}

unsigned ticks(void)
{
    /* returns timer ticks (approx. 18.2/sec)
    using only lower registers */

    union REGS in, out;

    in.x.ax = 0x0;
    int86(0x10, &in, &out);
    return(out.x.dx);
}

```

```

    }
    int timeout(unsigned start, int delay)
    {
        /*      used for timing events of up to approx. 1 hour.
           used in conjunction w/ticks()      */

        unsigned current;

        current = ticks();
        if (start <= current && (start + delay) < current)
            return(1);
        if (start > current && (start - 65535 + delay) < current)
            return(1);
        return(0);
    }

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

Copyright Telefind Corporation 1990

```
/* mark the end of the command line you built, so you can add ending
   delimiter */
sys_command[i] = NULL;
/* add the ending quote for the users message so shell wont
   interepert special characters */
strcat(sys_command, "\\");
/* execute command you built */
system(sys_command);

printf("sending message: %s\n", sys_command);

}
else {
    if(strlen(mesg) == 0 ) {
        return(0);
    }
    /* print error for invalid message length */
    printf("telemail error: invalid message length: %s\n", mesg);
    return(0);
}

return(i);
}

/*****
 *
 * function: getline(hold-buffer, input-file-pointer)
 * arguments: pointer to buffer where line read will be heald,
 *             file pointer to input file
 * description: reads 1 line of text from the input line and stores the
 *              line read into the buffer passed.
 * returns: -1 if EOF or number of characters read in
 *
 *****/
getline(buff, fp)
char *buff;
FILE *fp;
{
    int ch, cnt;

    /* keep on reading characetr from file so long as end of file not
       reached or char is the end of line */
    for(cnt = 0; ((ch = fgetc(fp)) != EOF) && ch != '\n'; cnt++) {
        /* MOD BY OT 11/29/90 convert tab to space */
        /* convert tabs to single space */
        if(ch == 9) {
            ch = ' ';
        }
        /* MOD BY OT 11/29/90 dont allow control char */
        /* only load in ascii characters */
        if(isprint(ch) != 0) {
            buff[cnt] = ch;
        }
        else {
            /* turn control characters to spaces */
            buff[cnt] = ' ';
        }
    }

    /* mark the end of the buffer you built */
    buff[cnt] = '\0';
}
```

```

/*****
*
*   function: send_mesg(message-pointer)
*   arguments: pointer to text message(capcode,text) to be sent
*   description: takes passed message text makes sure the first 8 positions
*               are numeric(capcode). it builds and executes the network
*               send command(netsend.sh) to sedn the message passed.
*   returns: 0 if not sent otherwise the number of characters sent out
*
*****/
int send_mesg(mesg)
char *mesg;
{
    char sys_command[700];
    int i;
    int ch;
    char *mesg_ptr;

    /* left justify the message passed to remove leading spaces */
    strljust(mesg, 512);
    /* trim off trailing blank spaces from the message */
    strtrim(mesg);

    /* make sure you have a capcode at least */
    if(strlen(mesg) > 8) {

        /* start to build the command to be executed to send message retrieved
        from the mail box */
        strcpy(sys_command, "netsend.sh ");

        /* loop while still more characters in the message */
        for(mesg_ptr = mesg, i = 11; *mesg_ptr != NULL; i++, mesg_ptr++) {

            /* make sure the first 8 positions of the message are numeric */
            if((i < 19) && (*mesg_ptr < '0' || *mesg_ptr > '9')) {
                printf("telemail error: invalid capcode: %s\n", mesg);
                return 0;
            }

            /* is the user didnt seperate capcode & message then insert a
            space into the command */
            if(i == 19 && *mesg_ptr != ' ') {
                sys_command[19] = ' ';
                i = 20;
            }

            /* enclose the users message with ' so shell wont interpret
            special characters */
            if(i == 20) {
                sys_command[20] = '\\';
                i = 21;
            }

            /* put the character from the message onto to the
            command to be executed*/
            sys_command[i] = *mesg_ptr;
        }
    }
}

```

```

/* since your just starting clear the message area */
memset(mesg, NULL, MAXMSGLEN);

/* keep on geting lines from the file until you reach end of file */
while(getline(buff, fp) != -1) {

    /* every mail message start with the word "From " */
    if(strncmp(buff, "From ", 5) == 0) {
        /* set flag telling you are currently going thru mail header
           so you dont add it to the message */
        in_header = 1;
        /* call routine to the last message if any exists */
        send_mesg(mesg);
        continue;
    }

    /* a mail header end with the following string */
    if(strncmp(buff, "Content-Length:", 15) == 0) {
        /* turn off flag so you know you are no longer in mail
           message header */
        in_header = 0;
        /* clear the old message since this is a new one */
        memset(mesg, NULL, MAXMSGLEN);
        continue;
    }

    /* if the line you are now reading in not part of the mail header
       add it to the message */
    if(in_header == 0) {
        strljust(buff, 512);
        strtrim(buff);
        /* make sure you dont add more than the message length */
        if( (strlen(buff) + strlen(mesg)) < MAXMSGLEN) {
            strcat(mesg, " ");
            strcat(mesg, buff);
        }
    }
} /* end of read line while */

/* send the last message in the file */
send_mesg(mesg);
}

```